ENCAPSULATED HARDWARE CONFIGURATION/CONTROL

The present invention relates to configuration and control of hardware subsystems.

As integration density increases, hardware subsystems become increasingly complex. Such subsystems typically require initialization at start up and configuration or reconfiguration during subsequent operation. Control of the subsystem during operation may also entail configuration or reconfiguration. Such initialization, configuration, reconfiguration and control are presently accomplished through system programming, which may be embodied in hardware, firmware or software. Such system programming is laborious, error prone and, from a programming perspective, unsafe.

Presently the system programmer must first read a detailed specification of the hardware subsystem. This specification is often very large and may be unclear, contradictory, out-of-date or erroneous. Next, the programmer typically encodes a Hardware Abstraction Language that translates numerous registers and defined bits referred to in the specification into alphanumeric descriptions that permit programmers to configure and control the hardware subsystem with names instead of digits. There are typically no tools to aid in this task. Hence, construction of the Hardware Abstraction Language largely involves a laborious process of substitution that is unappealing to many programmers and hence error-prone.

As a result of this process, errors may occur in sequencing the commands and data passed through the Hardware Abstraction Interface. Errors may also occur because the specification was wrong or misunderstood. Moreover, because the hardware subsystem's interface is global in nature (i.e., any task typically has access to any subsystem's registers or memory), an errant or malicious task can wreak havoc. In this sense, the present scheme for configuration and control of complex hardware subsystems is unsafe.

What is needed is a technique for configuration and control of complex hardware subsystems that relieves the foregoing burden placed on the system programmer and that is by comparison safe and error-free.

The present invention, generally-speaking, provides a technique for configuring and controlling complex hardware subsystems that relieves the burden placed on the system programmer and that is, by comparison to present methods, safe and error-free. In accordance with one aspect of the invention, configuration of a hardware subsystem is accomplished by providing in hardware a configuration controller including a controller portion and a storage portion storing configuration parameters. The configuration controller is activated, for example in response to a Configuration/Control ID, and thereupon performs configuration of the hardware subsystem, including storing at least one configuration parameter in a register of the hardware subsystem. Typically, the configuration controller hardware and storage are embedded within the hardware subsystem to be configured or controlled. The configuration/control functionality is thus "encapsulated" within the hardware subsystem itself such that the system programmer need not be concerned with details of the functionality. The configuration controller may take the form of a state machine, for example.

A hardware subsystem may assume different configurations at different times during operation of the system. In this instance, multiple Configuration/Control IDs may be provided, each Configuration/Control ID corresponding to a particular configuration.

The invention will be better understood upon reference to the following detailed description and accompanying drawing. In the drawing:

Figure 1 is a circuit diagram of a system using encapsulated hardware configuration/control.

Referring now to Figure 1, there is shown a circuit diagram of a system using encapsulated hardware configuration/control. The system is an electronic system that may be of any of various forms, including for example, an electronic chassis, a circuit board,. a multi-chip module or other module, a "System-On-a-Chip" (SOC) integrated circuit or other integrated circuit, etc. The system is assumed to include a processor 101 and a system bus 103. The system bus may connect to other busses (not shown) using one or more bus bridge interfaces as necessary.

Within the system, various subsystems are connected to the system bus, including a subsystem 110 to be configured or controlled. In the context of SOC, for example, one such typical subsystem might be a USB controller, for example. Other typical subsystems might include, for example, an SDRAM controller, a PLL/clock subsystem block, etc.

The subsystem 110 to be configured or controlled is illustrated in simplified form as including various hardware registers 111 and including in addition a Configuration/Control State Machine 113 and associated memory 115. The state machine 113 may be patterned after a complex instruction set processor. The memory 115 will typically be read-only. The Configuration/Control State Machine 113 is in communication with the system bus, and the memory 115 is in communication with various registers, e.g., via a subsystem bus 117. Preferably, the width of the memory 115 matches the width of the subsystem bus 117.

In operation, to configure or control the subsystem 110, a single simple Configuration/Control ID is passed from the processor 101 to the Configuration/Control State Machine 113, which may appear to system software simply as a register. The Configuration/Control State Machine 113 responds to the Configuration/Control ID by causing one or more write cycles to the subsystem's register set 111 to be performed. The system programmer may be oblivious to the inner workings of the Configuration/Control State Machine 113.

Multiple sets of configuration data may be stored within different portions of the memory 115, allowing for different configurations or different control operations. Different Configuration/Control IDs may be used to designate different sets of configuration data or different control operations.

One example of the use of multiple Configuration/Control IDs for a single subsystem is a USB (Universal Serial Bus) block in which particular ports (i.e., endpoints) can operate in Control, Interrupt, Isochronous or Bulk mode. In this instance, a different Configuration/Control ID is assigned to each mode. For each different mode, the Configuration/Control State Machine 113 responds to the corresponding Configuration/Control ID to write to potentially a large number of registers (e.g., endpoint enable register, endpoint interrupt register, DMA control register, etc.) without the significant potential for error encountered in the prior art approach. System software does not need to be aware of any of these "internal" registers; instead, system software simply needs to be aware of a relatively few Configuration/Control IDs. By writing a single Configuration/Control ID to a single "register" (the Configuration/Control State Machine 113), the desired configuration is reliably achieved.

A Configuration/Control ID may enable or disable various subsystem hardware options. In the case of a USB controller, for example, a control function might be to enable or disable automatic re-initialization of the DMA controller after a USB packet has been successfully received. (When this option is enabled, the DMA function is ready to accept the next packet without software intervention.) A Configuration/Control ID corresponding to Control mode would typically enable this option, whereas a Configuration/Control ID corresponding to Bulk mode typically would not.

Multiple different Control IDs may be used to access certain specific inner workings of a subsystem. For example, in a USB subsystem, one function that is controlled by the software is to put the subsystem into Suspend Mode in order to reduce power when there is no USB bus traffic. A single unique Control ID may be assigned to this function. Software would simply write this ID to the subsystem in order to enable this mode. The actual register and the particular bit in the register that performs this function would be hidden from view (i.e. encapsulated). An added benefit of this scheme is that a complete redesign to the USB block could be implemented without change to the software.

The invention having thus been described with respect to specific embodiments, it will be appreciated by those of ordinary skill in the art that the invention can be embodied in other specific forms without departing from the spirit or essential character thereof. The presently disclosed embodiments are therefore considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated by the appended claims, rather than the foregoing description, and all changes which come within the meaning and range of equivalence thereof are intended to be embraced therein.